Informe de análisis sobre tarea 102

Integrantes del grupo C1.026:

* Ignacio Blanquero Blanco ([ignblabla@alum.us.es](mailto:ignblabla@alum.us.es))
* Adrián Cabello Martín ([adrcabmar@alum.us.es](mailto:adrcabmar@alum.us.es))
* María de la Salud Carrera Talaverón ([marcartal1@alum.us.es](mailto:marcartal1@alum.us.es))
* Joaquín González Ganfornina ([joagongan@alum.us.es](mailto:joagongan@alum.us.es))
* Natalia Olmo Villegas ([natolmvil@alum.us.es](mailto:natolmvil@alum.us.es))

Fecha: Sevilla, 18 de Marzo 2024

Tabla de contenido

[Resumen del Informe 3](#_Toc164527887)

[Historial de Versiones 4](#_Toc164527888)

[Introducción 5](#_Toc164527889)

[Informe de Análisis – Acme-SF-D02 6](#_Toc164527890)

[Registro de Análisis 1 6](#_Toc164527891)

[Requisito 6](#_Toc164527892)

[Análisis y decisiones 6](#_Toc164527893)

[Validación por el profesor 7](#_Toc164527894)

[Registro de Análisis 2 8](#_Toc164527895)

[Requisito 8](#_Toc164527896)

[Análisis y decisiones 8](#_Toc164527897)

[Validación por el profesor 9](#_Toc164527898)

[Registro de Análisis 3 10](#_Toc164527899)

[Requisito 10](#_Toc164527900)

[Análisis y decisiones 10](#_Toc164527901)

[Validación por el profesor 10](#_Toc164527902)

[Registro de Análisis 4 11](#_Toc164527903)

[Requisito 11](#_Toc164527904)

[Análisis y decisiones 11](#_Toc164527905)

[Validación por el profesor 11](#_Toc164527906)

[Registro de Análisis 5 12](#_Toc164527907)

[Requisito 12](#_Toc164527908)

[Análisis y decisiones 12](#_Toc164527909)

[Validación por el profesor 12](#_Toc164527910)

[Registro de Análisis 6 13](#_Toc164527911)

[Requisito 13](#_Toc164527912)

[Análisis y decisiones 13](#_Toc164527913)

[Validación por el profesor 13](#_Toc164527914)

[Conclusiones 14](#_Toc164527915)

[Bibliografía 15](#_Toc164527916)

# Resumen del Informe

El objetivo de este informe es proporcionar detalles sobre el análisis de todos los tipos de requisitos especificados en esta segunda entrega del proyecto, así como las decisiones tomadas para llevarlos a cabo y enlazarlo, si procediera, a las validaciones y observaciones planteadas por el profesor encargado de la asignatura.

# Historial de Versiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Contenidos** | **Fecha** | **Contribuyente** |
| V0.1 | Estructura inicial Analysis Report D02 | 18/03/2024 | Natalia Olmo Villegas |
| V1.1 | Análisis completo de los requisitos | 20/04/2024 | Natalia Olmo Villegas |

# Introducción

En esta tercera entrega teníamos que implementar tres requisitos funcionales obligatorios y otros tres requisitos funcionales suplementarios. La carga de trabajo ha aumentado notablemente, por tanto, resulta de vital importancia una buena gestión de las tareas a cumplimentar por parte del Project Manager, Natalia Olmo Villegas en este caso. A continuación, se describirán con más detalle todos los requisitos implementados individualmente por Natalia Olmo Villegas.

# Informe de Análisis – Acme-SF-D02

Antes de comenzar con el análisis cabe destacar que al comienzo de la implementación de todos estos requisitos por parte de Natalia Olmo Villegas surgieron varios problemas, debido a que este integrante del grupo todavía no estaba familiarizado con el framework. No obstante, tras la implementación de todos ellos se ha logrado que Natalia Olmo Villegas pueda trabajar de forma fluida sin grandes dificultades.

## Registro de Análisis 1

### Requisito

Operaciones de los clientes en contratos:

* Listar los contratos que ellos han creado.
* Mostrar los detalles de sus contratos.
* Crear, actualizar, o borrar sus contratos. Los contratos pueden ser actualizados o borrados siempre y cuando no hayan sido publicados. Para que un contrato sea publicado, el sumatorio de todos los presupuestos no puede exceder del coste total correspondiente al proyecto.

### Análisis y decisiones

Por una parte, se ha decidido crear un servicio por cada funcionalidad que se pedía implementar tal y como se aconsejó en clase: mostrar, listar, crear, actualizar, borrar y publicar contratos.

A la hora de listar los contratos se ha supuesto que no sería necesario mostrar todos los atributos por cuestiones de diseño, por lo que se ha decidido que sólo se mostrarían los atributos "code", "providerName", "customerName". A su vez, no se ha impuesta ninguna restricción adicional para poder acceder a este listado, sólo que esos contratos pertenecieran al cliente logueado.

Con respecto al servicio de crear contratos, es de gran importancia comentar que se ha añadido a la entidad Contract un atributo “draftMode”, que nos permitirá conocer cuando un contrato está en modo borrador o no. Si se encuentra en modo borrador, podrá ser editado. En esta clase, dentro del método validate() se ha comprobado que el presupuesto no sea negativo y que no sea mayor que el coste del proyecto, ya que ese nos requería implementar esta regla de negocio en los requisitos de la primera entrega, pero no podíamos hacerlo hasta ahora debido a nuestra falta de conocimiento en ese momento.

A la hora de actualizar un contrato se decidió que esta acción se podría llevar a cabo siempre y cuando el contrato no sea nulo, se encuentre en modo borrador y pertenece al cliente logueado. Las validaciones y el resto del código es prácticamente igual que el de crear. Para eliminar un contrato lo más destacable sería que se ha especificado en el método perform() que una vez que se elimina un contrato también desaparecen sus informes de progresos asociados.

Por último, al publicar un contrato se valida la condición de que todos los presupuestos de los contratos asociados a un proyecto en concreto no deben sobrepasar el coste del proyecto y en el método perform() se deberá poner el draftMode a false para que no pueda ser editado.

En cuanto a decisiones de diseño del frontend se ha decidido que en los formularios a la hora de elegir el proyecto asociado a un contrato se abra un desplegable con todos los proyectos disponibles y que sean públicos. Con respecto a los botones, todos los botones aparecerán mientras el contrato esté en modo borrador, mientras que sólo el botón de los informes de progreso asociados aparecerá cuando esté publicado.

### Validación por el profesor

El profesor revisó este requisito en la primera sesión de laboratorio y detectó un fallo de validación en el servicio que se encargaba de actualizar los contratos: al darle directamente al botón de actualizar sin haber cambiado ningún dato, no se mostraba ningún mensaje de error y se permanecía en la página. Más tarde, Natalia Olmo Villegas consiguió arreglar este problema, lo que estaba ocurriendo era que estaba detectando que el código de contrato ya estaba en uso y además no se había asignado ningún mensaje de error a esa excepción. Por tanto, se reformuló la condición de validación contemplando la posibilidad de que se guardara el mismo código de contrato que ya tenía asignado ese mismo contrato para que no ocurriera ningún error.

Asimismo, se planteó la duda de si el cliente a la hora de crear un contrato debía acceder a todos los proyectos o se debía tener en cuenta alguna restricción. El profesor se encargó de aclarar esta duda indicando que sólo tendría sentido que se accedieran a todos los proyectos públicos.

## Registro de Análisis 2

### Requisito

Operaciones de los clientes en informes de progreso:

* Listar los informes de progreso de sus contratos.
* Mostrar los detalles de sus informes de progreso.
* Crear y publicar un informe de progreso.
* Actualizar o borrar un informe de progreso siempre y cuando no se haya publicado.

### Análisis y decisiones

Por una parte, se ha decidido crear un servicio por cada funcionalidad que se pedía implementar tal y como se aconsejó en clase: mostrar, listar, crear, actualizar, borrar y publicar informes de progreso. Este último no se pedía en los requisitos, pero se ha decidido añadir ya que más adelante hay un requisito que consiste en listar todos los informes de progreso de un contrato publicado. Este listado es accesible a todos los usuarios autenticados, así que estos sólo deberían acceder a informes de progreso que ya han sido publicados y sin posibilidad de modificación.

A la hora de listar los informes de progreso se ha supuesto que no sería necesario mostrar todos los atributos por cuestiones de diseño, por lo que se ha decidido que sólo se mostrarían los atributos "responsiblePerson", "completeness". A su vez, se ha decidido que se va a mostrar el botón de los informes de progreso asociados a ese contrato una vez que el contrato ha sido publicado, ya que si todavía no se han decretado con seguridad todos los detalles del contrato y no ha sido publicado no tendría sentido que tuviera informes de progreso.

Con respecto al servicio de crear informes de progreso, es de gran importancia comentar que se ha añadido a la entidad ProgressLog un atributo “draftMode”, que nos permitirá conocer cuando un informe de progreso está en modo borrador o no. Si se encuentra en modo borrador, podrá ser editado. En esta clase, dentro del método validate() se ha comprobado que el recordId sea único y se ha establecido el modo borrador a verdadero en el método load() y en el método perform()

A la hora de actualizar un informe de progreso se decidió que esta acción se podría llevar a cabo siempre y cuando el contrato al que está asociado no sea nulo y el mismo no se encuentre en modo borrador, pertenece al cliente logueado y que el informe de progreso esté en modo borrador. Las validaciones y el resto del código es prácticamente igual que el de crear.

Por último, al publicar un informe de progreso, tras hacer la misma comprobación que para actualizar y eliminar para ver si se tiene atorización para realizar esta acción a través del método authorise(), se realizan las validaciones oportunas, que son las mismas que las de los métodos anteriores, y por último en el método perform() se deberá poner el draftMode a false para que no pueda ser editado.

En cuanto a decisiones de diseño del frontend se ha seguido la misma estructura que para contrato y no ha surgido ningún problema.

### Validación por el profesor

En un primero momento se podía crear y acceder a los informes de progreso estuviera o no el contrato en modo borrador, pero el profesor revisó este requisito en la segunda sesión de revisión de laboratorio y señaló que los informes de progreso sólo podrían ser accedidos y modificados si el contrato estaba publicado. Asimismo, surgió el mismo problema que había ocurrido a la hora de actualizar los contratos: al darle directamente al botón de actualizar el informe de progreso sin haber cambiado ningún dato, no se mostraba ningún mensaje de error y se permanecía en la página. Por lo tanto, se tomaron las acciones correspondientes que se tomaron con contrato, pero esta vez con los informes de progreso.

## Registro de Análisis 3

### Requisito

Operaciones de los clientes en la pantalla principal:

* Mostrar las pantallas principales de los clientes.

### Análisis y decisiones

Se definieron los métodos correspondientes que se pedían en los requisitos de la entrega anterior en el repositorio y por consiguiente en su correspondiente servicio y controlador, sin encontrar ningún problema. Se decidió que en el repositorio al llamar a estos métodos habría que introducir cómo parámetro el id del cliente logueado para poder recoger los datos acerca de ese cliente en concreto y no tener en cuenta todos los clientes de la base de datos.

### Validación por el profesor

El profesor, tras revisar este requisito en la segunda sesión de revisión del proyecto de laboratorio, indicó que se debían filtrar todos estos métodos por el cliente que estuviera logueado, ya que en un primer momento no se había contemplado esta restricción y se hacían las operaciones requeridas con todos los clientes registrados en la base de datos.

## Registro de Análisis 4

### Requisito

Operaciones por usuarios anónimos en cuentas de usuario:

* Registrarse en el Sistema y convertirse en un cliente.

### Análisis y decisiones

Dado que para realizar las acciones que se piden el usuario debe estar autenticado, dentro del paquete authenticated se ha creado otro paquete client y se ha creado un repositorio, un controlador y dos servicios, uno de ellos para crear al cliente y otro para actualizarlo, que se pide en el siguiente requisito. Para ello se ha seguido el modelo de consumer y provider, que venían proporcionados por el proyecto starter Hello World.

No ha surgido ningún problema grave durante el desarrollo de este más allá de que no se sabía cómo implementar una barra de opciones para elegir el tipo de cliente, lo que después se acabó averiguando. Esto se ha implementado en el método unbind() de ambos servicios con SelectChoices.

Para que esta acción de convertirse en un cliente a partir de otro usuario registrado se llevara a cabo se tuvo que añadir un par de líneas de código a la clase menú.jsp para indicar que habría un nuevo desplegable de nombre “Become a client”.

### Validación por el profesor

En un primer momento, como he comentado anteriormente, se había implementado el tipo de cliente como una caja de texto para que al crear o al actualizar el cliente se debería hacer escribiendo un tipo de cliente o el otro, que en este caso podría ser INDIVIDUAL o COMPANY. No obstante, en la segunda sesión de revisión de laboratorio, el profesor indicó que esto era muy poco intuitivo para el usuario de la aplicación y tras investigar cómo se podía programar para que aparecieran las 2 opciones posibles y no tener que escribirlas manualmente, se implementó mejorando la experiencia de la interfaz de usuario.

## Registro de Análisis 5

### Requisito

Operaciones de los clientes en sus cuentas de usuario:

* Actualizar sus perfiles.

### Análisis y decisiones

Teniendo en cuenta lo mencionado para el requisito anterior, este se ha implementado igual.

La única diferencia notable es que en el servicio se ha contemplado la posibilidad de que al actualizar se tratara del mismo código del cliente que se tenía antes y no debería de saltar error. Asimismo, a la hora de acceder a este formulario de actualización de perfil e hará a través del desplegable configurado en menú.jsp, pero esta vez, como se accede desde el cliente logueado, el tecto se mostrará será “Client data” en vez de “Become a Client”

### Validación por el profesor

El profesor revisó este requisito en la primera sesión de revisión de laboratorio y no encontró ningún problema.

## Registro de Análisis 6

### Requisito

Operaciones en contratos por cualquier usuario:

* Listar los contratos que se han publicado en el Sistema.
* Mostrar los detalles de los contratos que ellos pueden listar (incluyendo sus informes de progreso)

### Análisis y decisiones

Se creó un paquete any donde se introdujo a continuación el paquete contract y el paquete progressLog y a su vez el controlador, repositorio y servicios de listar y mostrar de cado uno de ellos respectivamente.

Aquí sólo se mostrarán los contratos que son públicos, es decir su atributo draftMode está a false, ya que no tiene sentido que todo el mundo pueda acceder a contratos que no se hayan acordado por completo y no se hayan publicado. Igualmente, se aplica el mismo razonamiento para los informes de progreso, por lo que sólo se mostrarán aquellos informes de progreso publicados y que a su vez pertenecen a un contrato publicado. Esto se ha conseguido a través de una consulta JPQL en el repositorio de los informes de progreso.

Por otra parte, en los listados de ambas no se han considerado todos los atributos ya que se suponía algo innecesario, por lo tanto, para listar los contratos se han elegido los atributos "code", "providerName", "customerName" y para listar los informes de progreso se han elegido los atributos "recordId", "completeness", "responsiblePerson"

En cuanto a las vistas del frontend, no se encuentra nada destacable que mencionar.

### Validación por el profesor

El profesor revisó este requisito en la primera sesión de revisión de laboratorio y no encontró ningún problema.

# Conclusiones

En esta tercera entrega, nos hemos enfrentado a un aumento significativo de la carga de trabajo al implementar varios requisitos funcionales de la tarea individual. La gestión efectiva de estas tareas ha sido dirigida por el Project Manager, que ha sido Natalia Olmo Villegas en esta ocasión.

La implementación de todos los requisitos asignados para esta entrega ha generado varios problemas al comienzo de esta debido a la falta de familiarización con el framework. No obstante, Natalia logró superar los requisitos satisfactoriamente.

Se tomaron decisiones de diseño tanto en el backend como en el frontend para optimizar la experiencia del usuario, como la inclusión de desplegables para facilitar la selección del tipo de cliente, mejorando así la usabilidad del sistema y la aparición o no del botón que redirige a los informes de progreso asociados a un contrato en función de si ese contrato está en modo borrador o no.

Los comentarios más destacados del profesor que contribuyeron a un mejor desarrollo de estos requisitos serían la identificación de errores en la validación de ciertas funcionalidades, como la actualización de contratos e informes de progreso, ya que no se estaba validando bien que el código de los contratos y de los informes de progreso debían ser únicos. Además, se proporcionaron aclaraciones sobre los requisitos, como la necesidad de filtrar los métodos por el cliente logueado a la hora de desarrollar el panel de control del cliente o que no se podía acceder a los informes de progreso de un contrato si este no había sido publicado. Asimismo, el profesor ofreció sugerencias para mejorar la experiencia del usuario, como la inclusión de opciones predefinidas en un desplegable en lugar de campos de texto para los tipos de cliente.

En resumen, se ha cumplido con una exitosa implementación de todos los requisitos necesarios para esta entrega y los comentarios del profesor en las sesiones de revisión han contribuido a una mejor compresión de los requisitos.

# Bibliografía

En blanco a propósito.